Graph Neural Networks in the Perspective of Text and Code Summarization

Tanvin Sharma
Jakov Mitrovski
Ahmad Owaydate

Abstract

In the evolving field of Natural Language Processing (NLP), Graph Neural Networks (GNNs) have become a key technology especially when it comes to summarizing text and code. This survey thoroughly examines the progress and applications of GNNs showcasing their impact on extractive and abstractive text summarization techniques. It also focuses on how GNNs have been integrated to address challenges in summarizing written content and programming code. It explores how GNNs have transformed the effectiveness of summarization processes, including their ability to handle multiple documents. Furthermore this survey delves into the use of GNNs in code summarization highlighting how they bridge the gap between programming languages and natural language comprehension. By providing an encompassing overview while pointing out research directions this paper positions GNNs as an important force behind future advancements, in NLP based summarization tasks.

ACM Reference Format:
Tanvin Sharma, Jakov Mitrovski, and Ahmad Owaydate. 2024. Graph Neural Networks in the Perspective of Text and Code Summarization. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 Introduction

There is no doubt that the amount of text available online is intimidating. At the same time, extensive textual repositories containing news articles, novels, books, legal documents, and more continue to expand exponentially on a daily basis. This growing corpus brings up a compelling necessity for users to efficiently integrate information. The need to consume information has created a crucial demand for the summarization and compression of textual resources.

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

The manual execution of this summarization process is undoubtedly costly and time-intensive. Consequently, these challenges have encouraged research in Automatic Text Summarization (ATS). The primary objective of this research field is to construct methodologies for generating summaries that contain the key points of input documents. This requires achieving a summary characterized by conciseness, accuracy, and fluency. Such automated summarization techniques aim to enable users to consume information from input documents without the need of excessive reading, thereby effecting substantial time and effort savings.

1.1 Why use Graph Neural Networks (GNNs)?

Despite significant advancements in the field of text summarization, it continues to face various challenges. GNNs in recent times have been successfully used in Natural Language Processing (NLP) tasks like classification [Liu et al. 2020b] and translation [Xu et al. 2020]. While GNNs might not resolve every issue in ATS, they can certainly offer a new perspective. In our opinion, the meaningful benefits GNNs provide to ATS are significant enough to justify research in this area.
Some key advantages of utilizing GNNs for NLP include:

- *Graph-based Representation of Text*: Sentences and documents can naturally be represented as graphs, where nodes represent sentences or words, and edges describe relationships or flows of information [Sonawane and Kulkarni 2014]. This representation aligns well with the structure of GNNs, allowing for more effective capture and analysis of the text's structure.
- *Capturing Contextual Relationships*: GNNs are adept at capturing the contextual relationships between different parts of the text as shown in [Wu et al. 2021]. By considering the interdependencies between sentences or phrases, GNNs can generate summaries that maintain the clarity and context of the original text.
- *Integration with Other Models*: GNNs can be integrated with other NLP models, such as BERT or LSTM Devlin et al. [2018]; Hochreiter and Schmidhuber [1997], to combine the strengths of different approaches. This is often used in encoder-decoder model types, where the GNN is used as an encoder and another model is used as a decoder LeClair et al. [2020]; Liu et al. [2020a]. This integration can lead to improved performance

in generating summaries that are both coherent and contextually relevant.

- *Scalability*: Various BERT based ATS models are computationally very complex as their complexity grows quadratically with the input length, due to the self attention mechanism [Vaswani et al. 2017]. This hinders its ability to deal with long text documents. GNNs on the other hand are comparatively simpler to scale and can be scaled to graphs containing thousands of nodes. Li et al. [Li et al. 2021] give insights as to how one can train large and deep GNNs.
- *Explainability*: Understanding why a model derived a certain conclusion is often a difficult task. This problem in the case of GNNs is fairly simplified. As shown in [Ying et al. 2019], the GNN Explainer can help the user understand which nodes were used by the model to reach its output.

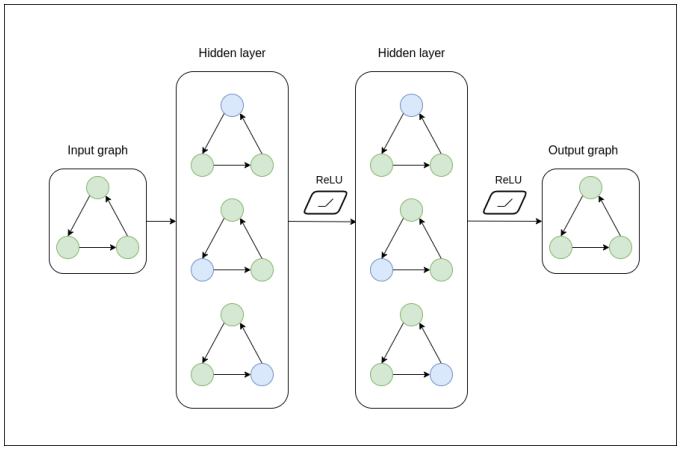

Figure 1. A basic GNN architecture

Text summarization has a range of applications, including code summarization [Haiduc et al. 2010], which is becoming increasingly important in the field of software development. As software projects become more complex, understanding and maintaining large codebases can be challenging de Souza et al. [2005]; Forward and Lethbridge [2002]; Roehm et al. [2012]. Code summarization helps by condensing source code into shorter, readable descriptions, making it easier for developers to understand the functionality of code segments without analyzing their full implementation.

The value of code summarization is highlighted in various areas within software engineering de Souza et al. [2005]; Forward and Lethbridge [2002]; Roehm et al. [2012]. With the rise of collaborative development and open-source projects, quickly grasping unfamiliar code is crucial. Code summarization provides descriptions of functions' source code, aiding developers to save time. It also plays a key role in automating documentation, enhancing code readability, and helping with software maintenance and debugging.

Code summarization differs from traditional text summarization due to the structured nature of programming languages. It requires a combination of NLP, machine learning, and software engineering techniques to effectively summarize code. This makes it not only a tool for simplifying code understanding but also a field that merges computer science precision with the intricacies of human language.

This paper aims to provide a comprehensive explanation of the role of GNNs in text summarization models, covering both extractive and abstractive methods. It will discuss the evolution of these techniques over time and the current advancements in the field. The integration of GNNs showcases a significant development in understanding and implementing summarization tasks, especially in dealing with the structural and relational aspects of text data. The paper will also delve into code summarization with particular attention to the application of GNNs in this subfield. The objective is to offer a thorough overview that informs software and machine learning engineers and serves as a resource for researchers interested in this growing area of study.

2 Background and Foundations

2.1 Graph Neural Networks (GNNs)

GNNs have emerged as a significant advancement in processing non-Euclidean data [Scarselli et al. 2008]. Unlike traditional neural networks, GNNs are adept at handling complex data structures that cannot be represented effectively in Euclidean space. This includes diverse domains like language, where textual data contains implicit structures that are not readily expressible in regular formats.

In GNNs, a graph is defined as G = (V, E), where V represents nodes and E denotes edges. Each node and edge carries feature vectors, encapsulating information about the structure and attributes of the graph. GNNs are versatile in managing both homogeneous and heterogeneous graphs. The latter is particularly important in scenarios where different types of data entities are present within the same graph. A basic GNN architecture is shown in Figure 1.

The processing in GNNs involves unique approaches like spatial convolution and message passing [Veličković 2022]. Spatial convolution extends the idea of convolution from Convolutional Neural Networks (CNNs) to irregular graph structures, allowing the aggregation of features from neighboring nodes. The message-passing mechanism facilitates the exchange of information between nodes across the graph, enabling the network to capture wider dependencies and relationships.

2.2 GNN Architecture, and Applications

2.2.1 Architecture. GNNs are characterized by their unique architecture suitable for processing graph-structured data.

They consist of layers where each layer aggregates information from neighboring nodes. Two prevalent architectures are Graph Convolutional Networks (GCNs) and Graph Attention Networks (GATs).

Graph Convolutional Networks (GCNs). Kipf et al. [Kipf and Welling 2016] introduced GCNs which extend traditional convolution to graph data, aggregating information from immediate neighbors. This involves summing transformed feature vectors of neighboring nodes, typically scaled by normalization factors like inverse square root of node degrees. This summation, in each convolutional layer, captures the features that rise from localized information. Deeper layers in GCNs aggregate information from an increasingly larger neighborhood.

Graph Attention Networks (GATs). GATs [Veličković et al. 2017] represent an advanced type of GNNs that incorporate the attention mechanism, a crucial concept from deep learning [Vaswani et al. 2017]. The key innovation in GATs is the dynamic weighting of the importance of each node's neighbors, in contrast to standard GNNs that typically aggregate neighbor information uniformly or based on static weights.

The attention mechanism in GATs allows each node to learn which neighbors to focus on during the feature aggregation phase. This process starts with a feature transformation where node features are linearly transformed using a shared weight matrix. Following this, the GAT layer calculates attention scores for each neighboring node pair. These scores are essentially scalar values that quantify the importance of a neighbor's features in the aggregation process. These attention scores are then normalized across all neighbors using a softmax function, allowing for a weighted average during feature aggregation.

GATs are proficient at handling non-uniform relationships within graphs, a common characteristic in many real-world scenarios. This capability to dynamically assign importance to different parts of the graph based on the learned attention scores makes GATs highly effective and versatile for a wide range of graph-based tasks. This attribute, coupled with their inherent ability to focus on the most relevant parts of the input graph, positions GATs as a powerful tool in the GNN landscape.

2.2.2 Applications.

Social Network Analysis. In social networks, GNNs can be used to understand and predict interactions and influences [Fan et al. 2019]. They can also identify influential nodes, predict link formation, and analyze community structures. Additionally, GNNs can capture the nuanced dynamics of social relationships and information flow, enhancing the accuracy of behavioral predictions and trend analysis in these networks.

Molecular Structure Analysis. In the field of chemistry, GNNs have emerged as a valuable tool for analyzing molecular structures [Jiang et al. 2021]. By representing molecules as graphs with atoms as nodes and bonds as edges, GNNs can predict molecular properties, drug interactions, and chemical reactions. This application is crucial for drug discovery, where understanding molecular interactions at a granular level is essential.

Recommendation Systems. GNNs improve recommendation quality by modeling complex user-item interactions within a network Fan et al. [2019]; Wu et al. [2022]. Their ability to capture the existing relationships between users and items leads to more personalized and accurate recommendations. Furthermore, GNNs can incorporate additional contextual information, such as user preferences or behavior, to refine the recommendation process.

Natural Language Processing (NLP). GNNs excel in capturing the inherent structure and relationships within textual data. They have been successfully applied to tasks such as document classification [Piao et al. 2022], entity recognition and relation extraction [Carbonell et al. 2021], semantic role labeling [Marcheggiani and Titov 2017] and code summarization LeClair et al. [2020]; Liu et al. [2020a]. GNNs are highly effective in modeling complex relationships in textual data, accurately capturing semantic connections between words, contextual dependencies, and the overall structure of documents. This capability allows GNNs to facilitate a deeper understanding of the varied semantic relationships among words and phrases. Thereby significantly improving the capability to create summaries that are aware of the context and coherence of the underlying text.

In summary, GCNs and GATs enable GNNs to efficiently process graph-structured data, making them versatile for a range of applications across different domains.

2.3 Extractive and Abstractive Text Summarization

Text summarization involves condensing a document's content into a shorter representation. Two main approaches are abstractive and extractive summarization. The abstractive summarization method generates a summary by paraphrasing and rephrasing the content in an original way, often using natural language generation techniques. Traditional abstractive summarization methods usually rely on rule-based systems, sentence compression or template based approaches [Andhale and Bewoor 2016]. This approach has some limitations including difficulty in handling complex language structures and generating coherent, contextually relevant summaries.

On the other hand, extractive summarization selects and extracts essential sentences or phrases directly from the source document to form the summary. Traditional methods involve statistical or machine learning models to identify

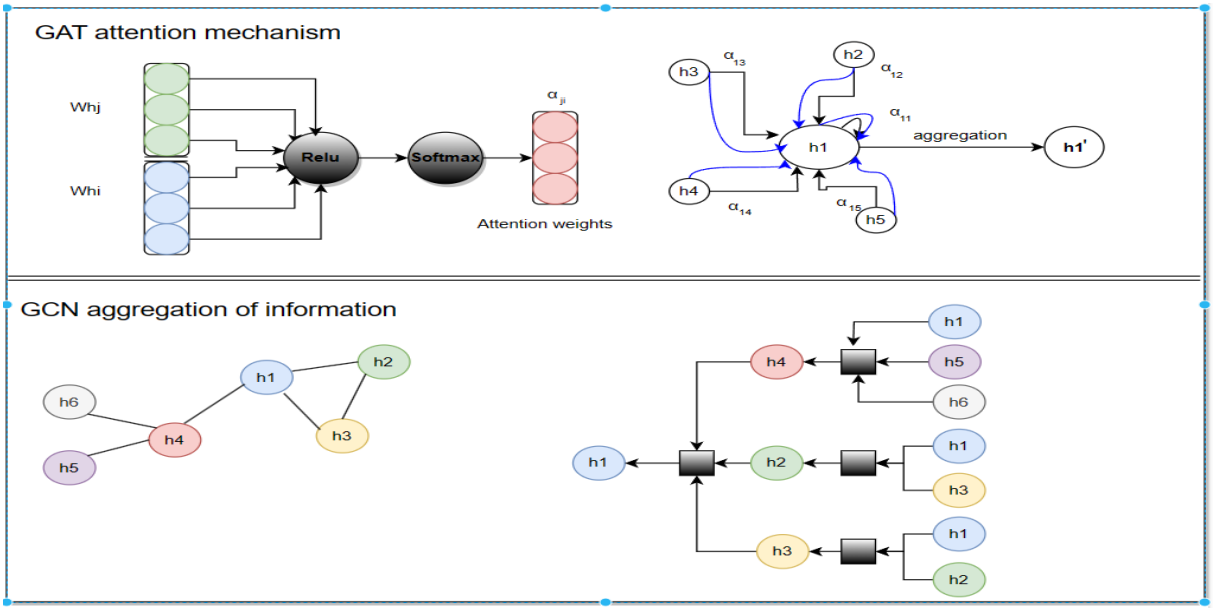

Figure 2. GCN aggregation of information(with k =2) vs GAT attention mechanism [Veličković et al. 2017]

important sentences or phrases. However, they may struggle with maintaining coherence and addressing redundancy [Allahyari et al. 2017].

2.4 Applying GNNs to Text Summarization

Graph-based methods for ATS, such as TextRank [Mihalcea and Tarau 2004], have been around since the early 2000s. For ATS, the challenge is converting implicit textual structures into explicit graph forms. This can be achieved by representing words and sentences as nodes in heterogeneous graphs, where different types of nodes can coexist. The design of these graphs can include directed or undirected edges to encode structural relationships, like the order of words or sentences.

In summary, GNNs for ATS effectively encode and process the complex, non-Euclidean structure of textual data, offering a powerful tool for both extractive and abstractive summarization tasks. Their flexibility and capability to handle heterogeneous graphs make them suitable for various summarization tasks, including single and multi-document summarization.

2.5 Code Summarization

Programming languages, unlike natural languages, are designed to be unambiguous and strictly structured, adhering to syntactical and semantic rules that computers can interpret. Each language, from high-level languages like Python and Java to low-level languages like C, presents unique features and constructs. The key to effective code summarization lies in understanding these constructs – such as loops, conditionals, classes, and functions – and the logic they encapsulate. This understanding is crucial for generating accurate and meaningful summaries that reflect the code's purpose and functionality.

Text summarization in NLP has evolved from simple extractive methods to more sophisticated abstractive techniques. This evolution is mirrored in code summarization, where the initial focus was on extracting comments or key lines of code. Recent advancements have shifted towards abstractive summarization, employing deep learning models to generate insightful, human-like summaries of code blocks.

Machine learning, particularly deep learning, has revolutionized the approach to code summarization. Models such as sequence-to-sequence (seq2seq) [Sutskever et al. 2014], transformers [Vaswani et al. 2017], and GNNS [Scarselli et al. 2008] have been adapted to understand the nuances of code. These models are trained on large datasets of source code and corresponding comments or documentation, learning to predict summaries that accurately reflect the code's functionality.

One of the primary challenges in code summarization is the need to balance the technical accuracy with the readability of the summaries. Additionally, the context in which a piece of code is used can significantly affect its interpretation, making it necessary for models to consider broader aspects of software projects. There is also the challenge of keeping up with the constantly evolving nature of programming languages and development paradigms.

The applications of code summarization are vast and growing. In addition to enhancing code readability and easing the

documentation process, it plays a significant role in educational tools, facilitating learning by providing clear explanations of code snippets. In the industry, it aids in code review processes, debugging, and understanding legacy systems, thereby improving the efficiency of software development and maintenance.

3 Extractive Summarization

Extractive summarization is based on the principle that relevant text from the original corpus can be extracted and organised as a summary. The training pipeline is therefore designed to select the most salient sentences from the input text and uses them to form the summary.

3.1 Graph based models

One of the most distinguished extractive summarization models, TextRank [Mihalcea and Tarau 2004] introduces the idea of building a graph where each sentence is connected to a node. Edges between these nodes are based on a similarity function and each node is initialised to a score of 1. Weighting of the graph is applied accordingly to the TextRank graph based ranking algorithm. The score is then recomputed for a fixed number of iterations (or until convergence). After this, the top scored sentences, decided by ranking them, are extracted as a summary. An extended version of this was implemented in LexRank [Erkan and Radev 2004] where the similarity function was modified to use Term Frequency - Inverse Document Frequency (TF-IDF) for sentence representations and the cosine similarity was calculated between these TF-IDF vectors. These methods, although using graphs, were not sophisticated enough as compared to models presented later. For example, LexRank had a common problem of not considering irregular words that are in one sentence and not in another. Also, long sentences were unfairly rewarded as they had a higher chance to match words against other sentences. It also did not consider words with naturally high frequencies like articles (a, an, the).

3.2 Neural models

Before 2020, there had been successful research in the use of deep neural networks on this task Liu and Lapata [2019]; Narayan et al. [2018]; Zhong et al. [2019]. BERTSum is a terrific example. It, and its variations, are used to benchmark new models that come out in the field of text summarization. To extract the sentences from a piece of text which are good contenders for the summary, a crucial step is to model cross-sentence relations. Most of the models of that time used Recurrent Neural Networks (RNNs) [Nallapati et al. 2017]. RNNs often struggle to effectively capture long-range dependencies [Bengio et al. 1993], presenting a challenge in accurately summarizing lengthy text. Using a graph structure was an intuitive way to perform this task, but it was difficult

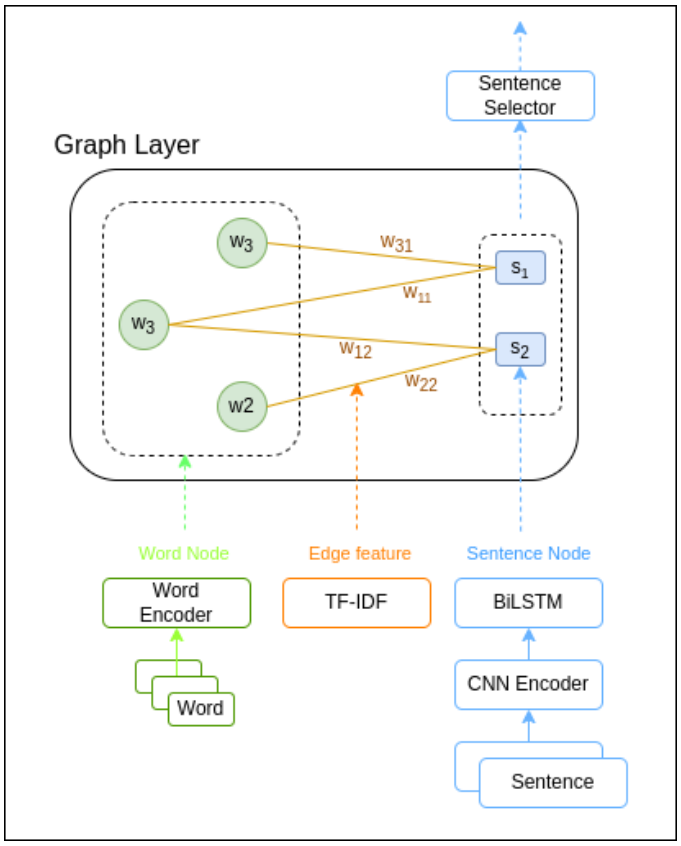

Figure 3. HSG Architecture [Wang et al. 2020]

to find such a structure which works. This led to the exploration of alternative approaches, such as the Transformer as proposed by Vaswani et al. [Vaswani et al. 2017]. The Transformer encoder, known for its ability to learn pairwise interactions between sentences, offers a promising direction for advancing text summarization techniques, as highlighted in the work of [Zhong et al. 2019].

3.3 HSG

Wang et al. [Wang et al. 2020] made an advancement in the field of extractive summarization by introducing a heterogeneous graph network called HeterSumGraph (HSG). Instead of making graphs with nodes on a sentence level, they introduced additional nodes - sentence nodes, word nodes and document nodes. The connections between these nodes are based on inclusion i.e. if a word corresponding to a word node, comes in a sentence, then its connected to the sentence node.

For the HeterSumGraph, feature vectors for nodes are obtained by encoders and the edge weights are obtained by calculating the TF-IDF score for each word. The network itself consists of a GAT layer which has been modified to consider the TF-IDF score values. After convolution, the hidden

state undergoes processing through a position-wise Feed-forward Neural Network (FFN), which comprises two linear transformations. The model employs three convolution layers: word-sentence, sentence-word, and word-document. It undergoes training on a node-based binary classification task. This task involves predicting whether to include a sentence node in the summary, a decision made by a single linear layer. This model architecture is represented in Figure 3. It has three major modules: graph initialisers, the heterogeneous graph layer and the sentence selector. The green circles and blue boxes represent the word and sentence nodes. The orange solid lines denote the TF-IDF score between word and sentence nodes with the corresponding weights. The final representation of sentence nodes will be used for summary selection.

This model brought impressive results. It outperformed non-BERT based models on both single and multi document summarization tasks.

An extension of this model, introduced by Jing et al. [Jing et al. 2021] encodes information into the graph. This model encodes the semantic and syntactical relationship between sentences within the graph. Antognini et al. [Antognini and Faltings 2019] also follow this idea of encoding information by an additional *universal feature vector* which is appended to each sentence node embedding. This vector is learned from a large unrelated corpus. This makes the model have a particular purpose as it focuses on the summarization of very small texts.

3.4 HaHSum

Jia et al. [Jia et al. 2020] introduced a model called Hierarchical Attentive Heterogeneous Graph for Text Summarization (HaHSum) which evolved from the previously mentioned idea. HaHSum models different levels of information including redundancy dependencies between sentences. The approach aims to iteratively refine the sentence representation with redundancy aware graph and delivers the label dependencies by message passing. The input graph is therefore more involved as it significantly reduces semantic sparsity. It uses three types of nodes, named entity, word and sentence nodes. The named entity nodes are anonymized tokens. Word nodes are connected with a directed edge to a sentence node if the word is contained within the sentence. Two name entity nodes are connected with an undirected edge if they represent the same entity. Finally, two sentence nodes are connected with an undirected edge if they share a trigram. On top of this, entities and words occurring in a sequence are connected with a directed edge.

HaHSum uses a GAT for all the five node type combinations and similarly to HSG, applies an FFN after the multi-headed attention. As a last layer, the model leverages a Fully Connected (FC) layer to perform binary classification. The architecture of HaHSum can be found in Figure 4

The results from HaHSum went on to prove that GNNs are a suitable choice for a task like this. It ended up outperforming extractive summarization models like MATCHSUM [Zhong et al. 2020] and even abstractive summarization models like PEGASUS [Zhang et al. 2020b].

4 Abstractive Summarization

Abstractive Summarization, often overshadowed by more robust extractive methods, tackles challenges rooted in the intricate nature of semantic representation, inference, and natural language generation.

4.1 Pre-GNN

Early models typically integrate an initial extractive preprocessing stage, manipulating the output for abstract formation Jing and McKeown [2000]; Knight and Marcu [2000]; Witbrock and Mittal [1999]. Before 2010, the field faced considerable challenges, with approaches falling into two groups: leveraging prior knowledge Aho et al. [1998]; De Jong et al. [1982] or employing Natural Language Generation (NLG) systems Jing and McKeown [2000]; Saggion and Lapalme [2002].

In 2010, Opinosis [Ganesan et al. 2010] emerged as a groundbreaking contribution to the field of abstractive summarization. Opinosis introduced a novel approach that departed from the prevalent methods of the time, demonstrating substantial improvements in agreement with human-generated summaries when compared to baseline methods.

In 2012, progress was marked by sentence simplification through monolingual machine translation [Wubben et al. 2012], showcasing promise in generating abstractive outputs. In 2015, Inspired by neural machine translation the paper by Rush et al. [Rush et al. 2015] combined a neural language model with a generation algorithm, though still falling short of human performance. After that in 2016, Chopra et al. [Chopra et al. 2016] introduced a conditional RNN architecture, extending abstractive summarization and addressing critical issues.

4.2 GNN in abstractive summarization

Then, the paper by Tan et al. [Tan et al. 2017] was published and it was a notable contribution to the field of abstractive summarization using GNNs. It distinguished itself by addressing the underexplored area of abstractive document summarization, using a novel graph-based attention mechanism within a hierarchical encoder-decoder framework. This approach specifically targets an important element: saliency, which in previous studies was neglected. The paper also introduces a hierarchical beam search algorithm to generate multi-sentence summaries, overcoming past challenges associated with long sequence processing. Also addressing difficulties in sequence-to-sequence models sets it apart from the rest before it. Experimental results demonstrate consistently better performance than previous models and competitive

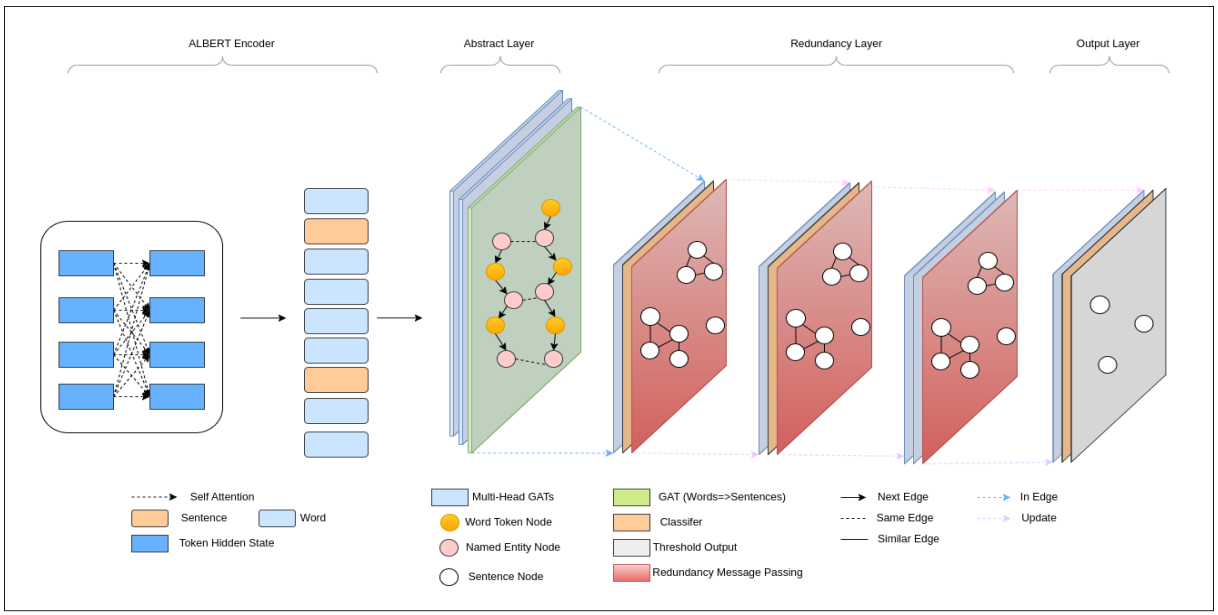

Figure 4. HaHSum Architecutre [Jia et al. 2020]

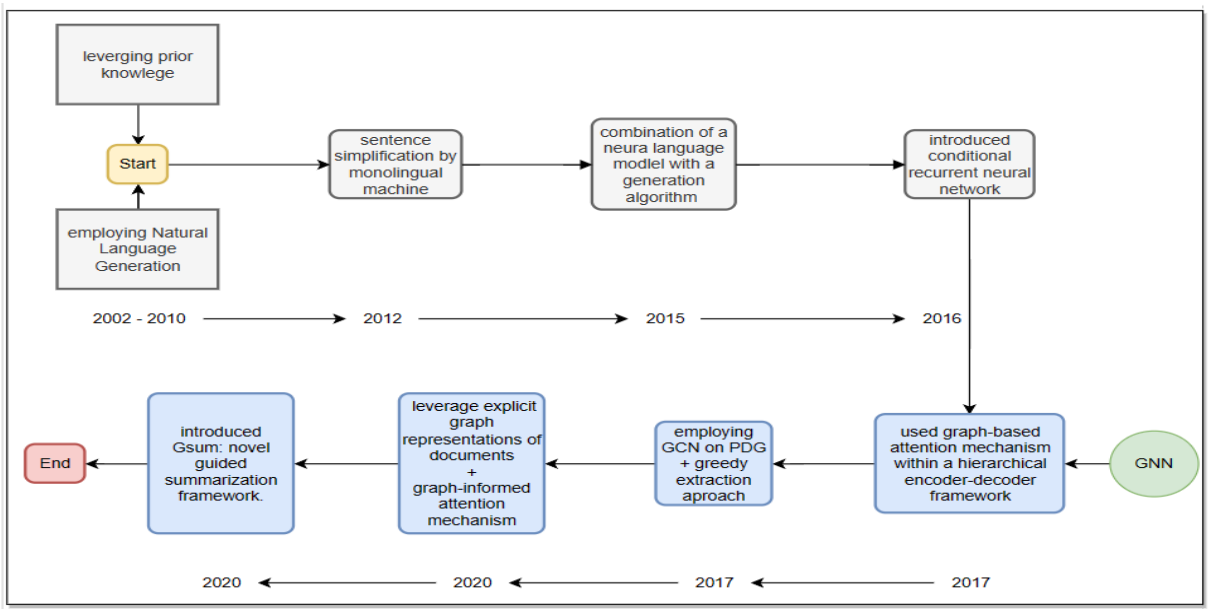

Figure 5. Abstractive timeline until the newest GNN development

results with state-of-the-art extractive methods. By tackling saliency effectively and taking a comprehensive approach, this paper significantly advanced the application of GNNs in abstractive summarization.

Recent developments in abstractive summarization have witnessed a transformative integration of GNNs, marking a huge shift in the field and bringing about significant advancements. The exploration of GNNs in this context introduces a powerful dimension, offering great capabilities that enhance various aspects of summarization.

Notably, the studies by Yasunaga et al. [Yasunaga et al. 2017] provide a pioneering approach to abstractive summarization, particularly inside the domain of multi-document processing. This paper distinguishes itself by the multi-document summarization system that it introduces, employing GCNs on a Personalized Discourse Graph (PDG). Unlike the RNN

models released before it, its innovative methodology captures the intricate sentence relations across multiple documents which leads to improved salience prediction and summarization. Also, the layer-wise propagation technique used in GCNs improves the generation of high-level hidden features while incorporating crucial graph information. Furthermore, due to the model's unique greedy extraction approach and its avoidance of redundancy gives it the ability to produce concise and non-repetitive abstractive summaries. Moreover, the aforementioned features, the clear advantage over traditional graph-based extractive summarization models, and competitive results with other state-of-the-art multi-document summarization systems, highlight the effectiveness of GNNs in abstractive summarization. This positions the study as a significant and innovative contribution to the field of neural summarization.

Li et al. [Li et al. 2020] further advanced GNN-based abstractive summarization by addressing key challenges in multi-document summarization (MDS). Their study leverages explicit graph representations of documents, such as similarity graphs and discourse graphs, to enhance the effectiveness of generating abstractive summaries and processing multiple input documents. This sets it apart from previous abstractive approaches that ignore the benefits of explicit graph structure. They developed an MDS model that takes into account both the document representation and summary generation processes. The model also introduced a graph-informed attention mechanism, that captures richer cross-document relations. These relations are captured during document encoding coupled with a hierarchical graph attention mechanism to guide the summary generation process effectively. This paper explored different graph representations and demonstrated substantial improvements in multi-document summarization performance. Additionally, it highlights the model's compatibility with pre-trained language models, like BERT, which leads to enhanced summarization results. this paper contributes by showing the effectiveness of graph modeling, proposing methods to integrate explicit graphs into the neural architecture, and demonstrating significant performance gains over strong baselines. Therefore, it offers a systematic and innovative approach to abstractive summarization, emphasizing the importance of explicit graph representations in multi-document scenarios and its capability to capture complex relationships within textual data.

Dou et al. [Dou et al. 2020] introduce a novel guided summarization framework named Gsum. Gsum is a framework that is general and extensible, which unlike studies before, focuses on specific type of guidance. This is represented by accommodating various external guidance signals during test time. This study proposes a model based on encoder-decoders, which when generating summaries, incorporates both source documents and guidance signals. The four types of guidance signals that are explored in this paper are highlighted sentences, keywords, salient relational triples, and retrieved summaries. This diversity of guidance allows for a comprehensive investigation into their complementarity. This paper evaluated the Gsum framework on six popular summarization benchmarks, achieving state-of-the-art performance, particularly when using highlighted sentences as guidance. This paper, not only demonstrates the effectiveness of the guided summarization framework, but also conducts in-depth analyses. It reveals the complementary nature of different guidance signals, showcasing the model's ability to generate more novel words, produce more faithful summaries, and offer controllability in the output by incorporating user-specified guidance signals. This diversity is a significant contribution to advancing the understanding and capabilities of guided neural abstractive summarization, showcasing the versatility and adaptability of GNNs in addressing specific requirements in summarization tasks.

Collectively, these recent developments underscore the growing role of GNNs in abstractive summarization, demonstrating their effectiveness in improving salience estimation, enhancing controllability, and contributing to state-of-the-art performance. As researchers continue to explore and refine GNN-based models, the field is likely to witness further innovations and breakthroughs in the pursuit of generating more accurate, coherent, and contextually rich abstractive summaries from textual content.

5 Code Summarization

5.1 Code Summarization in the Scope of Abstractive and Extractive Text Summarization

While traditional text summarization is primarily divided into extractive and abstractive methods, applying these categories directly to code summarization is not straightforward, as indicated by Eddy et al. [2013]; Haiduc et al. [2010]; LeClair et al. [2019]; McBurney and McMillan [2015]. The challenge arises from the complexity and structured nature of programming languages, which differ significantly from natural language. Understanding and translating complex programming constructs like loops, conditionals, and function calls, into a coherent summary that reflects the code's purpose and functionality, requires a more specialized approach. Although extractive and abstractive methods provide a starting point, they often fall short in adequately addressing the requirements of code summarization. This urges the development of more specific computational models that can interpret the delicate structure of code.

In 2010, Haiduc et al. [Haiduc et al. 2010] introduced one of the first key contributions for summarizing code extractively. This marked a shift towards utilizing text retrieval techniques and Latent Semantic Indexing (LSI) to identify and extract keywords from the source code. These keywords formed the basis of the summary. The foundational work

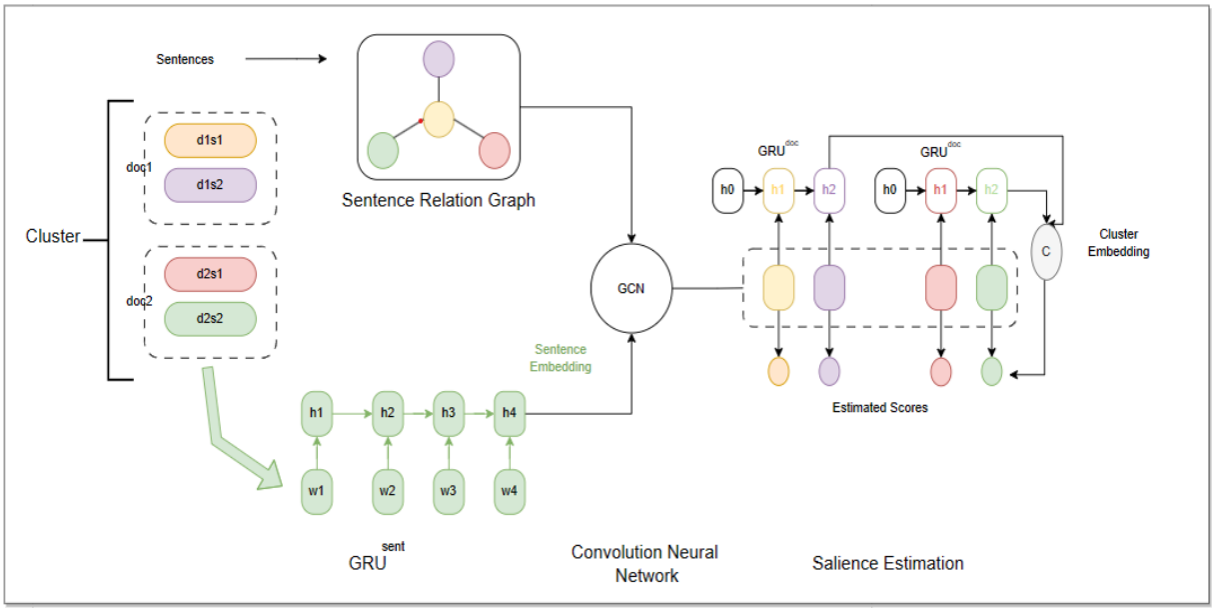

Figure 6. Illustration of the architecture for sentence salience estimation. [Yasunaga et al. 2017]

of Haiduc et al. set the stage for later advances in extractive code summarization techniques, including the use of TF-IDF, LSI, and Latent Dirichlet Allocation (LDA) [Haiduc et al. 2010]. The mentioned methods focus on assembling crucial elements from the source code to construct a useful summary.

Recent advancements have been made towards generation-based (abstractive in our context) methods to enhance the generalization performance in code summarization. Iyer et al. and Luong et al. Iyer et al. [2016]; Luong et al. [2015] have proposed studies that investigate Sequence-to-Sequence (Seq2Seq) architectures for producing source code summaries. Hu et al. and Alon et al. Alon et al. [2018]; Hu et al. [2018] devised methods that process the source code or often its abstract syntax tree (AST) as a sequence. The proposed Seq2Seq architecture in this case employs an encoder-decoder model which also utilizes an attention mechanism for summary generation. However, these approaches heavily depend on the sequential nature of this architecture, which struggle to capture the complex semantics in the source code, such as long-term control and data dependencies [Bengio et al. 1993]. These methods, primarily designed for NLP, struggle with the unique challenges of code summarization and demand a different approach.

5.2 Encoding and Enhancing through GNNs

GNNs, with their capability to model relationships and dependencies within structured data, emerge as particularly suited for tackling the challenges of code summarization. The integration of GNNs into traditional models, as shown by DISCOBERT [Xu et al. 2019], enhances their performance by leveraging the encoding capabilities of GNNs. This integration is favourable in code summarization where the inherent structure of code can be efficiently captured and processed.

Allamanis et al. [Allamanis et al. 2017] introduced a novel approach in the field by representing programs as graphs, significantly advancing the understanding of source code semantics. Building on this, studies like those by Fernandes et al. [Fernandes et al. 2018] leverage GNNs to capture more complex code structures such as control flow and program dependencies.

Typically, these works involve converting code into a graph-structured format for preprocessing, which is then processed by GNNs, to derive node and graph embeddings Allamanis et al. [2017]; Fernandes et al. [2018]. However, a common limitation in most GNN-based encoders is their restriction to message passing within a k-hop neighborhood, as highlighted by Zhao et al. Zhao and Akoglu [2019]. This limits their capacity to only local neighborhood information, neglecting the global interactions among nodes.

Li et al. [Li et al. 2019] used deep GCNs to address this challenge. However, the computational complexity often becomes restrictive, especially when large and complex program are in question.

5.3 GNNs in Action for Code Summarization

Firstly, we understand how GNNs improve code summarization by grasping the semantics, translating to natural language, and maintaining context relevance. Next, let's explore specific cases from recent studies that demonstrate

these concepts in practice, highlighting the field's practical uses and advancements.

Xu et al. [Xu et al. 2018] developed a model named Graph2seq. Following the standard encoder-decoder architecture, the graph encoder first produces node embeddings for each node. Utilizing this result, it then creates graph embeddings as shown in Figure 7. Secondly, there is a sequence decoder that takes a few different inputs, namely the node and graph embeddings, it employs attention over the node embeddings and generates a resulting sequence that represents the summary. They applied this model to convert SQL queries into natural language queries, utilizing forward and backward propagation over the graph and incorporating node-level attention. They achieved state-of-the-art results in translating SQL to natural language using the BLEU-4 metric. Their study also investigated the impact of the number of hops in the model, finding that while different hop numbers converge to similar results, some models could achieve comparable performance with fewer hops, reducing computational demands.

LeClair et al.[LeClair et al. 2020] modifies Xu et al. Graph2seq approach [Xu et al. 2018] and publish a study that introduces an advanced neural model architecture to enhance automatic source code summarization. As shown in Figure 8, this approach integrates GCNs with conventional sequence-based methods, creating a dual-encoder system. One encoder processes the source code sequence using a Gated Recurrent Unit (GRU) layer [Cho et al. 2014], while the other leverages GCNs for modeling the Abstract Syntax Tree (AST) of the code. This combination allows the model to capture both the sequential nature of the source code and the complex structural information contained in the AST. An attention mechanism then highlights significant tokens, which aids in the prediction of the next token in the summary sequence. By employing GCNs to process the AST, the model enables the nodes to learn rich representations based on their neighboring nodes, thus enhancing the model's comprehension of the code's structure in relation to the source code tokens. Validated on a dataset of 2.1 million Java method-comment pairs, their approach shows substantial improvement over several baselines such as Alon et al. [2018]; Xu et al. [2018]. The study not only demonstrates the efficiency of their model but also provides insightful analysis into how the incorporation of GCNs contributes to the advancement of source code summarization.

Liu et al. [Liu et al. 2020a] presented the Hybrid GNN (HGNN) framework, a novel approach for automatic code summarization that also incorporates GNNs. As depicted in Figure 9, HGNN is designed in a way that it combines the advantages of both retrieval-based and generation-based methods through a unique retrieval-augmented mechanism. The framework utilizes GNNs to process a Code Property Graph (CPG), derived from the source code, and an attention-based dynamic graph to effectively capture local and global structural dependencies within the code. This process involves a hybrid message-passing strategy across the graph representations. As a last component, to generate the summary, the model has an attention-based LSTM decoder. To assess HGNN's efficiency, the team introduced a new benchmark dataset from a variety of extensive open-source C projects, including over 95,000 unique functions. This method demonstrated superior performance over existing methods such as results achieved in Fernandes et al. [2018]; Iyer et al. [2016]; Wan et al. [2018]; Zhang et al. [2020a]. This model demonstrated state-of-the-art results in BLEU-4, ROUGE-L, and METEOR metrics, thereby showing its effectiveness in translating complex source code structures into coherent natural language summaries.

6 Related Work

The use of GNNs in Text and Code Summarization is an interesting area of research that hasn't been extensively studied. While there is a recent review by Malekzadeh et al. [Malekzadeh et al. 2021] focusing on graph-based text classification, it's important to note that there are numerous surveys available for each of these individual fields.

6.1 GNN Surveys

Wu et al. [Wu et al. 2020] proposed a survey which is among the most notable ones on GNNs. This survey specializes in GNNs, imparting a taxonomy that categorizes them into recurrent GNNs, convolutional GNNs, graph autoencoders, and spatial-temporal GNNs. The article delves into applications of GNNs throughout diverse domain names, with a selected emphasis on their application in NLP.

Building upon the previously mentioned survey, Wu et al. [Wu et al. 2022] evaluate and expand on ideas related to GNNs. The survey consists of a deeper exploration of packages and further facts, in particular in the context of NLP.

Wu et al. [Wu et al. 2023] introduced a study which serves as the first complete evaluation of GNNs in only NLP.

6.2 NLP Summarization Surveys

For Code Summarization, Zhang et al. [Zhang et al. 2022] give an in depth survey on Automatic Source Code Summarization era. This survey explores numerous methods and applications employed in automatically summarizing code.

In the area of ATS, Gambhir et al. [Gambhir and Gupta 2017] offer an intensive exploration of various methods and strategies. The survey gives a complete review of the modern-day ATS. Furthermore, Widyassari et al. [Widyassari et al. 2022] provide a broad and systematic review of research in the field of text summarization published from 2008 to 2019.

Expanding in addition to preceding studies, El et al. [El-Kassas et al. 2021] contribute to the sphere of ATS with a

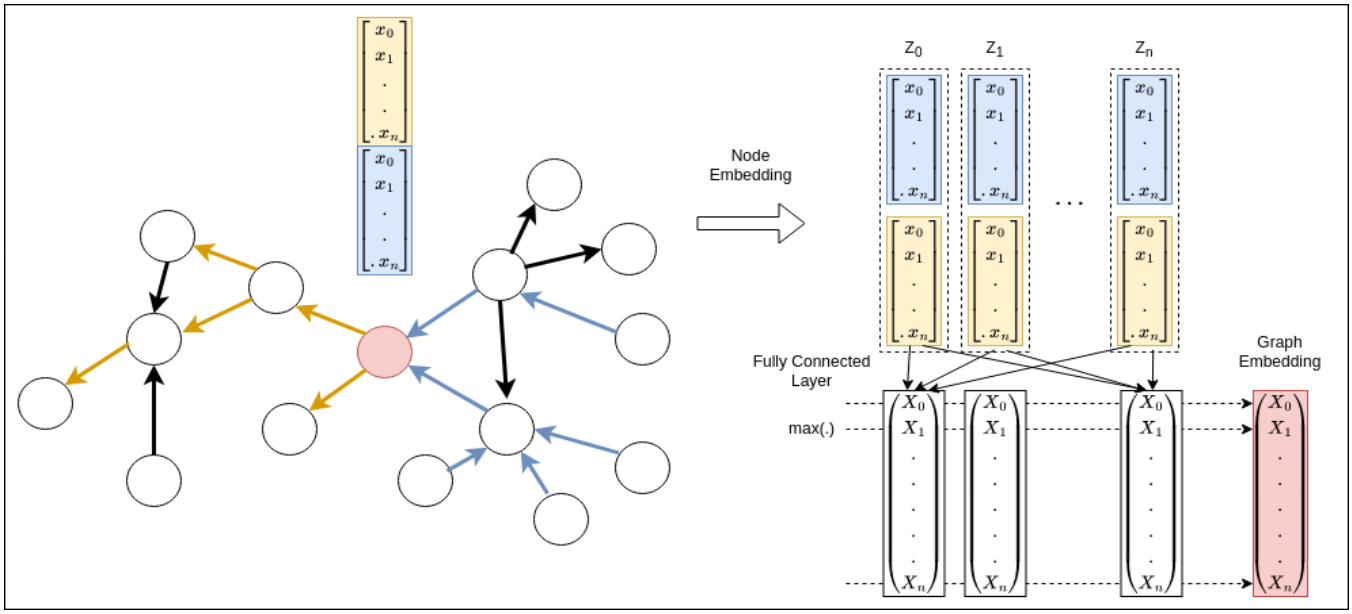

Figure 7. The Graph2Seq encoder model [Xu et al. 2018].

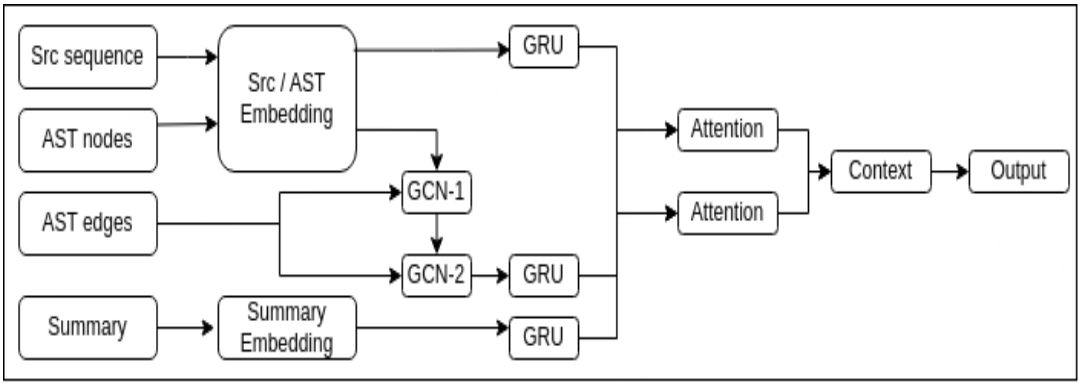

Figure 8. A high level overview of the model architecture proposed by [LeClair et al. 2020].

survey that covers conventional methods but additionally explores more recent programs and improvements in automated textual content summarization.

This synthesis of surveys gives a nuanced understanding of the landscape of GNNs and ATS, paving the manner for more focused and knowledgeable research in those domains.

6.3 Contribution of our survey

There are not enough up-to-date surveys and attention on the field of summarization and therefore the contributions of our survey are as follows:

- We provide a detailed exploration of the role of GNNs in text summarization, covering both extractive and abstractive methods.
- We discuss the historical development of abstractive and extractive text summarization techniques, highlighting the evolution of methodologies and current advancements in the field, with a specific focus on the integration of GNNs.
- The paper also addresses code summarization, as an interesting subfield of natural language summarization, emphasizing the application of GNNs in this subfield.

7 Conclusion

The evolution of extractive summarization models has progressed from early graph-based approaches like TextRank to sophisticated structures such as HeterSumGraph and HaHSum. These advancements highlight the growing impact of GNNs in addressing the challenges of summarizing diverse

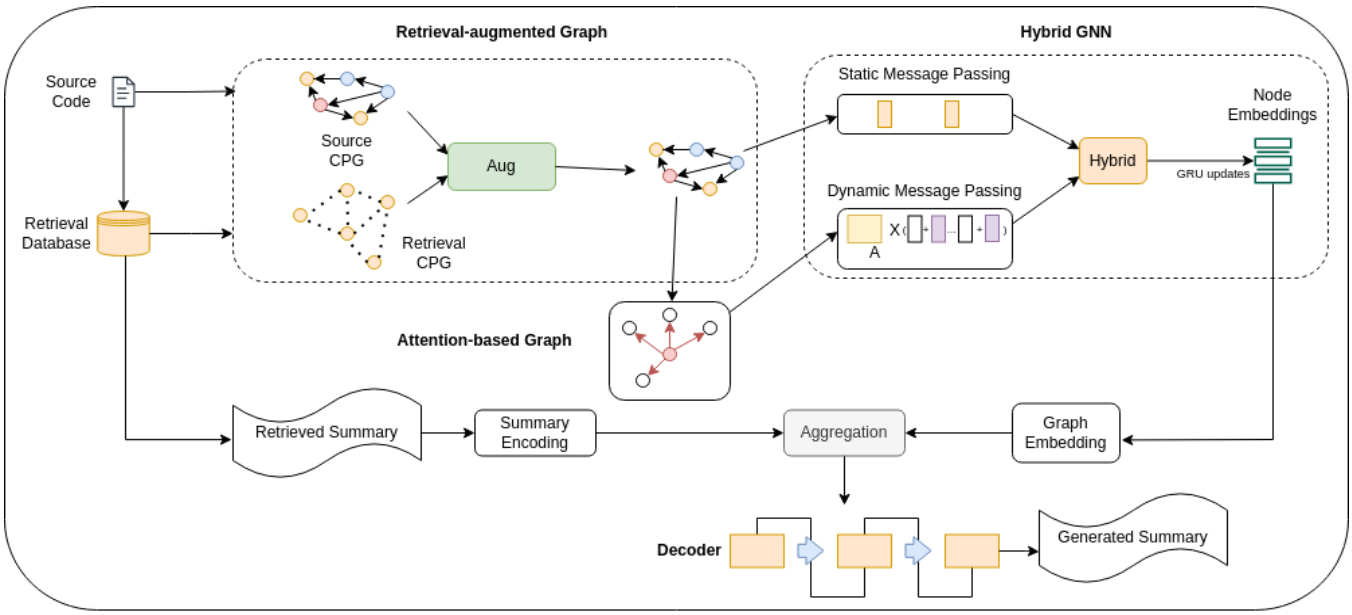

Figure 9. An overview of the model architecture proposed by [Liu et al. 2020a].

textual content, marking a promising direction for future innovations in ATS.

Using Graph Neural Networks (GNNs) in abstractive summarization is a significant advancement, solving challenges and pushing the field forward. Key contributions include involve the adaptability of GNNs in capturing saliency, enhancing multi-document summarization, and introducing inventive guided frameworks such as Gsum. These advancements highlight the growing importance of GNNs in shaping abstractive summarization, paving the way for future innovations and more precise summaries from text.

The part of this survey on GNNs in NLP for code summarization highlights their key role in improving this area. Innovations like graph2seq and the improved GNN frameworks demonstrate GNNs' ability to effectively bridge the gap between programming languages and natural language. Significant improvements in standard metrics and the introduction of new benchmarks mark the progress in this field.

In conclusion, GNNs are starting to become an influential force in advancing NLP-based text and code summarization, indicating a promising direction for future research and development.

References

Alfred V Aho, Shih-Fu Chang, Kathleen R McKeown, Dragomir R Radev, John R Smith, and Kazi A Zaman. 1998. Columbia digital news project: an environment for briefing and search over multimedia information. *International Journal on Digital Libraries* 1 (1998), 377–385.

Mehdi Allahyari, Seyedamin Pouriyeh, Mehdi Assefi, Saeid Safaei, Elizabeth D Trippe, Juan B Gutierrez, and Krys Kochut. 2017. Text summarization techniques: a brief survey. *arXiv preprint arXiv:1707.02268* (2017).

Miltiadis Allamanis, Marc Brockschmidt, and Mahmoud Khademi. 2017. Learning to represent programs with graphs. *arXiv preprint arXiv:1711.00740* (2017).

Uri Alon, Shaked Brody, Omer Levy, and Eran Yahav. 2018. code2seq: Generating sequences from structured representations of code. *arXiv preprint arXiv:1808.01400* (2018).

Narendra Andhale and L.A. Bewoor. 2016. An overview of Text Summarization techniques. In *2016 International Conference on Computing Communication Control and automation (ICCUBEA)*. 1–7. https://doi.org/10.1109/ICCUBEA.2016.7860024

Diego Antognini and Boi Faltings. 2019. Learning to create sentence semantic relation graphs for multi-document summarization. *arXiv preprint arXiv:1909.12231* (2019).

Yoshua Bengio, Paolo Frasconi, and Patrice Simard. 1993. The problem of learning long-term dependencies in recurrent networks. In *IEEE international conference on neural networks*. IEEE, 1183–1188.

Manuel Carbonell, Pau Riba, Mauricio Villegas, Alicia Fornés, and Josep Lladós. 2021. Named entity recognition and relation extraction with graph neural networks in semi structured documents. In *2020 25th International Conference on Pattern Recognition (ICPR)*. IEEE, 9622–9627.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078* (2014).

Sumit Chopra, Michael Auli, and Alexander M Rush. 2016. Abstractive sentence summarization with attentive recurrent neural networks. In *Proceedings of the 2016 conference of the North American chapter of the association for computational linguistics: human language technologies*. 93–98.

David De Jong, Roger A Morse, and George C Eickwort. 1982. Mite pests of honey bees. *Annual Review of Entomology* 27, 1 (1982), 229–252.

Sergio Cozzetti B de Souza, Nicolas Anquetil, and Káthia M de Oliveira. 2005. A study of the documentation essential to software maintenance. In *Proceedings of the 23rd annual international conference on Design of communication: documenting & designing for pervasive information*. 68–75.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).

Zi-Yi Dou, Pengfei Liu, Hiroaki Hayashi, Zhengbao Jiang, and Graham Neubig. 2020. Gsum: A general framework for guided neural abstractive summarization. *arXiv preprint arXiv:2010.08014* (2020).

Brian P Eddy, Jeffrey A Robinson, Nicholas A Kraft, and Jeffrey C Carver. 2013. Evaluating source code summarization techniques: Replication and expansion. In *2013 21st International Conference on Program Comprehension (ICPC)*. IEEE, 13–22.

Wafaa S El-Kassas, Cherif R Salama, Ahmed A Rafea, and Hoda K Mohamed. 2021. Automatic text summarization: A comprehensive survey. *Expert systems with applications* 165 (2021), 113679.

G. Erkan and D. R. Radev. 2004. LexRank: Graph-based Lexical Centrality as Salience in Text Summarization. *Journal of Artificial Intelligence Research* 22 (dec 2004), 457–479. https://doi.org/10.1613/jair.1523

Wenqi Fan, Yao Ma, Qing Li, Yuan He, Eric Zhao, Jiliang Tang, and Dawei Yin. 2019. Graph neural networks for social recommendation. In *The world wide web conference*. 417–426.

Patrick Fernandes, Miltiadis Allamanis, and Marc Brockschmidt. 2018. Structured neural summarization. *arXiv preprint arXiv:1811.01824* (2018).

Andrew Forward and Timothy C Lethbridge. 2002. The relevance of software documentation, tools and technologies: a survey. In *Proceedings of the 2002 ACM symposium on Document engineering*. 26–33.

Mahak Gambhir and Vishal Gupta. 2017. Recent automatic text summarization techniques: a survey. *Artificial Intelligence Review* 47 (2017), 1–66.

K Ganesan, SK Raza, and R Vijayaraghavan. 2010. Chemical warfare agents. *Journal of pharmacy and bioallied sciences* 2, 3 (2010), 166.

Sonia Haiduc, Jairo Aponte, Laura Moreno, and Andrian Marcus. 2010. On the use of automated text summarization techniques for summarizing source code. In *2010 17th Working conference on reverse engineering*. IEEE, 35–44.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.

Xing Hu, Ge Li, Xin Xia, David Lo, and Zhi Jin. 2018. Deep code comment generation. In *Proceedings of the 26th conference on program comprehension*. 200–210.

Srinivasan Iyer, Ioannis Konstas, Alvin Cheung, and Luke Zettlemoyer. 2016. Summarizing source code using a neural attention model. In *54th Annual Meeting of the Association for Computational Linguistics 2016*. Association for Computational Linguistics, 2073–2083.

Ruipeng Jia, Yanan Cao, Hengzhu Tang, Fang Fang, Cong Cao, and Shi Wang. 2020. Neural extractive summarization with hierarchical attentive heterogeneous graph network. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 3622–3631.

Dejun Jiang, Zhenxing Wu, Chang-Yu Hsieh, Guangyong Chen, Ben Liao, Zhe Wang, Chao Shen, Dongsheng Cao, Jian Wu, and Tingjun Hou. 2021. Could graph neural networks learn better molecular representation for drug discovery? A comparison study of descriptor-based and graph-based models. *Journal of cheminformatics* 13, 1 (2021), 1–23.

Baoyu Jing, Zeyu You, Tao Yang, Wei Fan, and Hanghang Tong. 2021. Multiplex graph neural network for extractive text summarization. *arXiv preprint arXiv:2108.12870* (2021).

Hongyan Jing and Kathleen McKeown. 2000. Cut and paste based text summarization. In *1st Meeting of the North American chapter of the association for computational linguistics*.

Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907* (2016).

Kevin Knight and Daniel Marcu. 2000. Statistics-based summarization-step one: Sentence compression. *AAAI/IAAI* 2000 (2000), 703–710.

Alexander LeClair, Sakib Haque, Lingfei Wu, and Collin McMillan. 2020. Improved code summarization via a graph neural network. In *Proceedings of the 28th international conference on program comprehension*. 184–195.

Alexander LeClair, Siyuan Jiang, and Collin McMillan. 2019. A neural model for generating natural language summaries of program subroutines. In *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*. IEEE, 795–806.

Guohao Li, Matthias Müller, Bernard Ghanem, and Vladlen Koltun. 2021. Training graph neural networks with 1000 layers. In *International conference on machine learning*. PMLR, 6437–6449.

Guohao Li, Matthias Muller, Ali Thabet, and Bernard Ghanem. 2019. Deepgcns: Can gcns go as deep as cnns?. In *Proceedings of the IEEE/CVF international conference on computer vision*. 9267–9276.

Wei Li, Xinyan Xiao, Jiachen Liu, Hua Wu, Haifeng Wang, and Junping Du. 2020. Leveraging graph to improve abstractive multi-document summarization. *arXiv preprint arXiv:2005.10043* (2020).

Shangqing Liu, Yu Chen, Xiaofei Xie, Jingkai Siow, and Yang Liu. 2020a. Retrieval-augmented generation for code summarization via hybrid gnn. *arXiv preprint arXiv:2006.05405* (2020).

Xien Liu, Xinxin You, Xiao Zhang, Ji Wu, and Ping Lv. 2020b. Tensor graph convolutional networks for text classification. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 34. 8409–8416.

Yang Liu and Mirella Lapata. 2019. Text summarization with pretrained encoders. *arXiv preprint arXiv:1908.08345* (2019).

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025* (2015).

Masoud Malekzadeh, Parisa Hajibabaee, Maryam Heidari, Samira Zad, Ozlem Uzuner, and James H Jones. 2021. Review of graph neural network in text classification. In *2021 IEEE 12th annual ubiquitous computing, electronics & mobile communication conference (UEMCON)*. IEEE, 0084–0091.

Diego Marcheggiani and Ivan Titov. 2017. Encoding sentences with graph convolutional networks for semantic role labeling. *arXiv preprint arXiv:1703.04826* (2017).

Paul W McBurney and Collin McMillan. 2015. Automatic source code summarization of context for java methods. *IEEE Transactions on Software Engineering* 42, 2 (2015), 103–119.

Rada Mihalcea and Paul Tarau. 2004. Textrank: Bringing order into text. In *Proceedings of the 2004 conference on empirical methods in natural language processing*. 404–411.

Ramesh Nallapati, Feifei Zhai, and Bowen Zhou. 2017. Summarunner: A recurrent neural network based sequence model for extractive summarization of documents. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 31.

Shashi Narayan, Shay B Cohen, and Mirella Lapata. 2018. Ranking sentences for extractive summarization with reinforcement learning. *arXiv preprint arXiv:1802.08636* (2018).

Yinhua Piao, Sangseon Lee, Dohoon Lee, and Sun Kim. 2022. Sparse structure learning via graph neural networks for inductive document classification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 36. 11165–11173.

Tobias Roehm, Rebecca Tiarks, Rainer Koschke, and Walid Maalej. 2012. How do professional developers comprehend software?. In *2012 34th International Conference on Software Engineering (ICSE)*. IEEE, 255–265.

Alexander M Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. *arXiv preprint arXiv:1509.00685* (2015).

Horacio Saggion and Guy Lapalme. 2002. Generating indicative-informative summaries with sumum. *Computational linguistics* 28, 4 (2002), 497–526.

Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. 2008. The graph neural network model. *IEEE transactions on neural networks* 20, 1 (2008), 61–80.

Sheetal S Sonawane and Parag A Kulkarni. 2014. Graph based representation and analysis of text document: A survey of techniques. *International Journal of Computer Applications* 96, 19 (2014).

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. *Advances in neural information processing*

systems 27 (2014).

Jiwei Tan, Xiaojun Wan, and Jianguo Xiao. 2017. Abstractive document summarization with a graph-based attentional neural model. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 1171–1181.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems* 30 (2017).

Petar Veličković. 2022. Message passing all the way up. *arXiv preprint arXiv:2202.11097* (2022).

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. *arXiv preprint arXiv:1710.10903* (2017).

Yao Wan, Zhou Zhao, Min Yang, Guandong Xu, Haochao Ying, Jian Wu, and Philip S Yu. 2018. Improving automatic source code summarization via deep reinforcement learning. In *Proceedings of the 33rd ACM/IEEE international conference on automated software engineering*. 397–407.

Danqing Wang, Pengfei Liu, Yining Zheng, Xipeng Qiu, and Xuanjing Huang. 2020. Heterogeneous graph neural networks for extractive document summarization. *arXiv preprint arXiv:2004.12393* (2020).

Adhika Pramita Widyassari, Supriadi Rustad, Guruh Fajar Shidik, Edi Noersasongko, Abdul Syukur, Affandy Affandy, et al. 2022. Review of automatic text summarization techniques & methods. *Journal of King Saud University-Computer and Information Sciences* 34, 4 (2022), 1029–1046.

Michael J Witbrock and Vibhu O Mittal. 1999. Ultra-summarization (poster abstract) a statistical approach to generating highly condensed non-extractive summaries. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*. 315–316.

Lingfei Wu, Yu Chen, Kai Shen, Xiaojie Guo, Hanning Gao, Shucheng Li, Jian Pei, Bo Long, et al. 2023. Graph neural networks for natural language processing: A survey. *Foundations and Trends® in Machine Learning* 16, 2 (2023), 119–328.

Shiwen Wu, Fei Sun, Wentao Zhang, Xu Xie, and Bin Cui. 2022. Graph neural networks in recommender systems: a survey. *Comput. Surveys* 55, 5 (2022), 1–37.

Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. 2020. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems* 32, 1 (2020), 4–24.

Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S. Yu. 2021. A Comprehensive Survey on Graph Neural Networks. *IEEE Transactions on Neural Networks and Learning Systems* 32, 1 (2021), 4–24. https://doi.org/10.1109/TNNLS.2020.2978386

Sander Wubben, Antal Van Den Bosch, and Emiel Krahmer. 2012. Sentence simplification by monolingual machine translation. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 1015–1024.

Jiacheng Xu, Zhe Gan, Yu Cheng, and Jingjing Liu. 2019. Discourse-aware neural extractive text summarization. *arXiv preprint arXiv:1910.14142* (2019).

Kun Xu, Lingfei Wu, Zhiguo Wang, Yansong Feng, Michael Witbrock, and Vadim Sheinin. 2018. Graph2seq: Graph to sequence learning with attention-based neural networks. *arXiv preprint arXiv:1804.00823* (2018).

Mingzhou Xu, Liangyou Li, Derek Wong, Qun Liu, Lidia S Chao, et al. 2020. Document graph for neural machine translation. *arXiv preprint arXiv:2012.03477* (2020).

Michihiro Yasunaga, Rui Zhang, Kshitijh Meelu, Ayush Pareek, Krishnan Srinivasan, and Dragomir Radev. 2017. Graph-based neural multi-document summarization. *arXiv preprint arXiv:1706.06681* (2017).

Zhitao Ying, Dylan Bourgeois, Jiaxuan You, Marinka Zitnik, and Jure Leskovec. 2019. Gnnexplainer: Generating explanations for graph neural networks. *Advances in neural information processing systems* 32 (2019).

Chunyan Zhang, Junchao Wang, Qinglei Zhou, Ting Xu, Ke Tang, Hairen Gui, and Fudong Liu. 2022. A survey of automatic source code summarization. *Symmetry* 14, 3 (2022), 471.

Jian Zhang, Xu Wang, Hongyu Zhang, Hailong Sun, and Xudong Liu. 2020a. Retrieval-based neural source code summarization. In *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering*. 1385–1397.

Jingqing Zhang, Yao Zhao, Mohammad Saleh, and Peter Liu. 2020b. Pegasus: Pre-training with extracted gap-sentences for abstractive summarization. In *International Conference on Machine Learning*. PMLR, 11328–11339.

Lingxiao Zhao and Leman Akoglu. 2019. Pairnorm: Tackling oversmoothing in gnns. *arXiv preprint arXiv:1909.12223* (2019).

Ming Zhong, Pengfei Liu, Yiran Chen, Danqing Wang, Xipeng Qiu, and Xuanjing Huang. 2020. Extractive summarization as text matching. *arXiv preprint arXiv:2004.08795* (2020).

Ming Zhong, Danqing Wang, Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2019. A closer look at data bias in neural extractive summarization models. *arXiv preprint arXiv:1909.13705* (2019).